# 图像空域滤波

## 简介

内容：图像的空域滤波
1、对图像进行平滑去噪（至少两种噪声，两种滤波器模板）
2、对图像进行锐化突出边缘等细节（至少两种边缘检测模板）

## 原理

（1） 均值去噪

设有一副$M \times M$的图像$f(x,y)$，若均值去噪后的图像为$g(x,y)$，则有：

$$g(x,y) = \frac{1}{N}\sum_{i,j\in s} f(i,j)$$

式中，$x,y = 0,1,\dots,M-1$，$s$为$(x,y)$领域内像素坐标的集合，$N$表示集合$s$内像素的总数

可见均值去噪就是将当前像素邻域内各像素的灰度平均值作为其输出值的去噪方法

（2） 超限像素平滑

对均值去噪稍加改进，可导出超限像素平滑法，它是将$f(x,y)$和邻域平均$g(x,y)$差的绝对值与选定的阈值作比较，根据比较结果决定点$(x,y)$的最后灰度$g\hat{}(x,y)$。其表达式为：

$$g\hat{}(x,y) = \begin{cases} g(x,y) & |f(x,y) - g(x,y)| > T \\ f(x,y) & |f(x,y) - g(x,y)| \le T \end{cases}$$

（3） 中值去噪

设有一副$M \times M$的图像$f(x,y)$，若平滑图像为$g(x,y)$，则有：

$$g(x,y) = med\left[\sum_{i,j\in s} f(i,j)\right]$$

式中，$x,y = 0,1,\dots,M-1$，$s$为$(x,y)$领域内像素坐标的集合，$med$表示取集合$s$内像素的中位数

可见中值去噪就是将当前像素邻域内各像素的灰度中位数作为其输出值的去噪方法

（4） 拉普拉斯算子锐化

拉普拉斯算子定义图像$f(x,y)$梯度为：

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

对于离散图像，其拉普拉斯算子为：

$$\nabla^2 f = \Delta_x^2 f(x,y) + \Delta_y^2 f(x,y)$$

其中

$$\Delta_x^2 f(x,y) = \Delta_x f(x+1,y) - \Delta_x f(x,y) = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\Delta_y^2 f(x,y) = \Delta_y f(x,y+1) - \Delta_y f(x,y) = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

因此，有

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

相当于原图像与模板$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$的卷积。

(5) Prewitt 算子和 Sobel 算子锐化

Prewitt 算子对应的模板为

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Sobel 算子对应的模板为

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

# MATLAB 代码

```
clc;clear;
image = rescale(double(imread('rice.png')));
figure(1);imshow(image);


%%
%添加高斯噪声并使用局部平滑法去噪
imageGaussianNoise1 = imnoise(image, 'gaussian');

%M*N 邻域平滑
M1 = 3;
N1 = 3;

imageGaussianDenoise1 = imfilter(imageGaussianNoise1, fspecial('average', [M1
, N1]));

```

```
figure(2);
subplot(1, 2, 1);imshow(imageGaussianNoise1, []); title('加高斯噪声');
subplot(1, 2, 2);imshow(imageGaussianDenoise1, []); title([num2str(M1), '*', num2str(N1), '邻域平滑']);


%%
%添加高斯噪声并使用超限像素平滑

M2 = 3;
N2 = 3;

imageGaussianNoise2 = imnoise(image, 'gaussian');
image_ = imfilter(imageGaussianNoise2, fspecial('average', [M2, N2]));
thresholdGaissian = 0.08;
imageGaussianDenoise2 = (abs(imageGaussianNoise2-image_) > thresholdGaissian) .* image_ + (abs(imageGaussianNoise2-image_) <= thresholdGaissian) .* imageGaussianNoise2;

figure(3);
subplot(1, 2, 1);imshow(imageGaussianNoise2, []); title('加高斯噪声');
subplot(1, 2, 2);imshow(imageGaussianDenoise2, []); title([num2str(M2), '*', num2str(N2), '邻域超限像素平滑（阈值', num2str(thresholdGaissian), ')']);


%%
%添加椒盐噪声并使用局部平滑法去噪
imageSaltpepperNoise1 = imnoise(image, 'salt & pepper');

%M*N 邻域平滑
M3 = 3;
N3 = 3;

imagesaltpepperDenoise1 = imfilter(imageSaltpepperNoise1, fspecial('average', [M3, N3]));

figure(4);
subplot(1, 2, 1);imshow(imageSaltpepperNoise1, []); title('加椒盐噪声');
subplot(1, 2, 2);imshow(imagesaltpepperDenoise1, []); title([num2str(M3), '*', num2str(N3), '邻域平滑']);


```

```
55.
56. %%
57. %添加椒盐噪声并使用超限像素平滑
58.
59. M4 = 3;
60. N4 = 3;
61.
62. imageSaltpepperNoise2 = imnoise(image, 'salt & pepper');
63. image_ = imfilter(imageSaltpepperNoise2, fspecial('average', [M4, N4]));
64. thresholdSaltpepper = 0.3;
65. imageSaltpepperDenoise2 = (abs(imageSaltpepperNoise2-
    image_) > thresholdSaltpepper) .* image_ + (abs(imageSaltpepperNoise2-
    image_) <= thresholdSaltpepper) .* imageSaltpepperNoise2;
66.
67. figure(5);
68. subplot(1, 2, 1);imshow(imageSaltpepperNoise2, []); title('加椒盐噪声');
69. subplot(1, 2, 2);imshow(imageSaltpepperDenoise2, []); title([num2str(M4), '*'
    , num2str(N4), '邻域超限像素平滑（阈值', num2str(thresholdSaltpepper), ')']);
70.
71.
72.
73. %%
74. %添加椒盐噪声并使用中值去噪
75. imageSaltpepperNoise3 = imnoise(image, 'salt & pepper');
76.
77. %M*N 窗口
78. M5 = 3;
79. N5 = 3;
80.
81. imageSaltpepperDenoise3 = medfilt2(imageSaltpepperNoise3, [M5, N5]);
82.
83. figure(6);
84. subplot(1, 2, 1);imshow(imageSaltpepperNoise3, []); title('加椒盐噪声');
85. subplot(1, 2, 2);imshow(imageSaltpepperDenoise3, []); title([num2str(M5), '*'
    , num2str(N5), '窗口中值去噪']);
86.
87.
88.
89. %%
90. %拉普拉斯算子锐化
91. a = 1;
92. laplacianMask = [0, 1 * a, 0;
93.         1 * a, 1 + (-4) * a, 1 * a;
94.         0, 1 * a, 0];
```

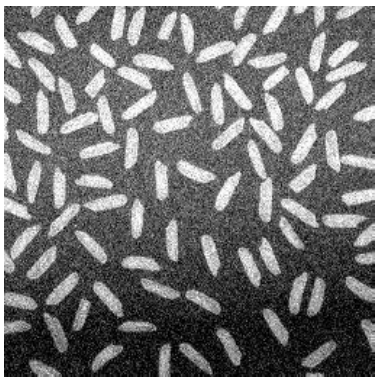
```
image_laplacian = imfilter(image, laplacianMask);
figure(7);imshow(image_laplacian, []);


%%
%水平 Prewitt 算子和 Sobel 算子锐化
prewittMask1 = fspecial('prewitt');
sobelMask1 = fspecial('sobel');
image_prewitt1 = imfilter(image, prewittMask1);
image_sobel1 = imfilter(image, sobelMask1);
figure(8);
subplot(1, 2, 1);imshow(image_prewitt1 + image, []);title('prewitt 水平边缘模板');
subplot(1, 2, 2);imshow(image_sobel1 + image, []);title('sobel 水平边缘模板');


%%
%垂直 Prewitt 算子和 Sobel 算子锐化
prewittMask2 = fspecial('prewitt')';
sobelMask2 = fspecial('sobel')';
image_prewitt2 = imfilter(image, prewittMask2);
image_sobel2 = imfilter(image, sobelMask2);
figure(9);
subplot(1, 2, 1);imshow(image_prewitt2 + image, []);title('prewitt 垂直边缘模板');
subplot(1, 2, 2);imshow(image_sobel2 + image, []);title('sobel 垂直边缘模板');
```

## 结果分析

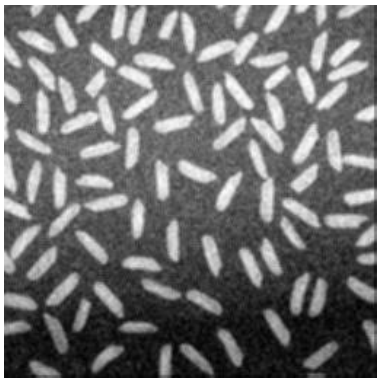
原图使用 rice.png

添加高斯噪声并使用均值去噪

加高斯噪声



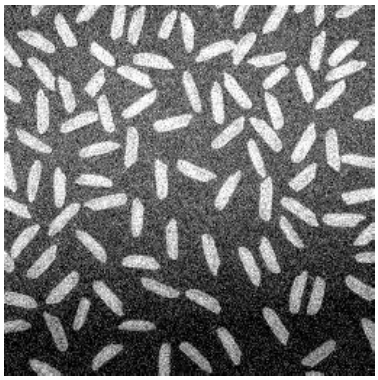
**3*3**邻域平滑



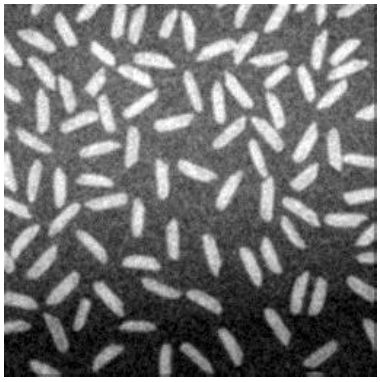
均值去噪对高斯噪声效果比较好，但图像大量细节丢失

添加高斯噪声并使用超限像素法去噪

加高斯噪声



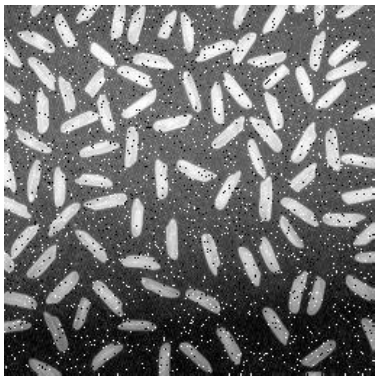
**3*3**邻域超限像素平滑（阈值 **0.08)**



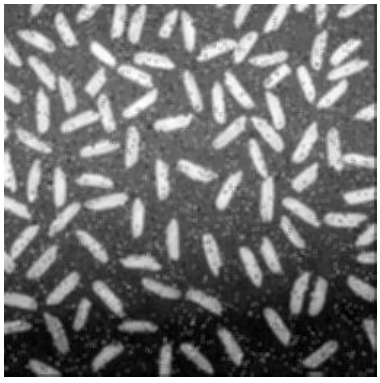
调节阈值可使去噪后的图像保留一定细节

接下来对付椒盐噪声
添加椒盐噪声并使用均值去噪

加椒盐噪声



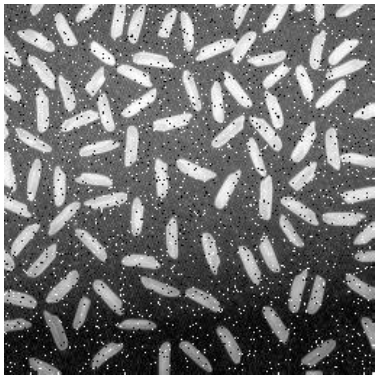
**3*3**邻域平滑



可以发现均值去噪对付椒盐噪声效果不太理想

添加椒盐噪声并使用超限像素平滑

加椒盐噪声

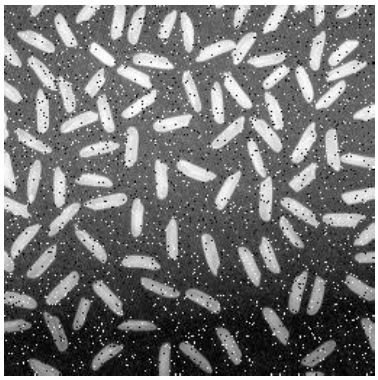


**3*3**邻域超限像素平滑（阈值 **0.3)**

调节阈值可以得到更好的效果

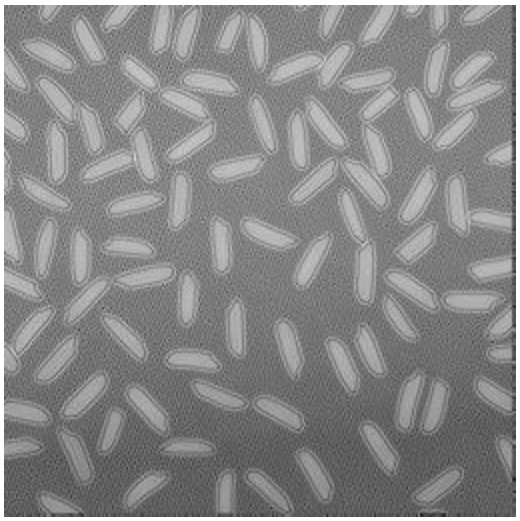
通常对付椒盐噪声比较好的方法是中值去噪，添加椒盐噪声并使用中值去噪

加椒盐噪声



**3*3**窗口中值去噪

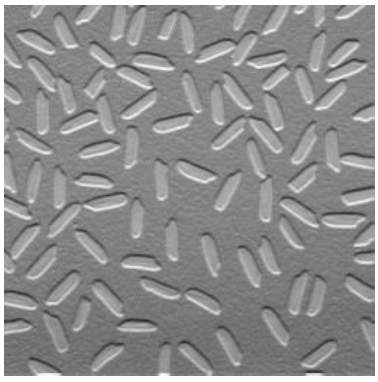
对图像平滑往往会丢失大量细节信息，因此需要对图像进行锐化处理
使用拉普拉斯算子对图像进行锐化处理



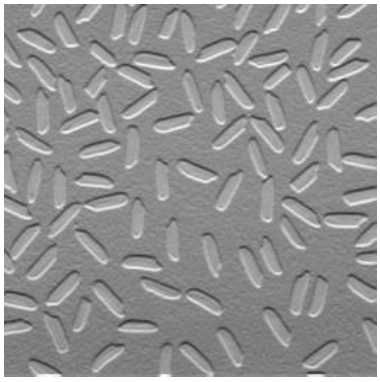
拉普拉斯算子边缘的方向信息被丢失，对孤立噪点的响应比像素线条的响应大

使用 prewitt 和 sobel 水平边缘模板对图像进行锐化处理

**prewitt** 水平边缘模板



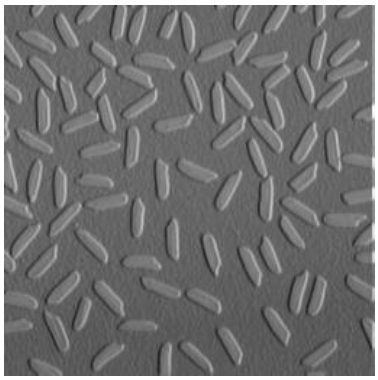
**sobel** 水平边缘模板



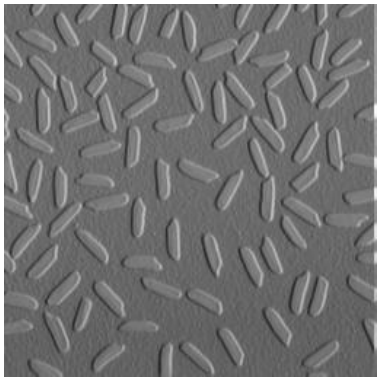
水平方向的边缘被增强

使用 prewitt 和 sobel 垂直边缘模板对图像进行锐化处理

**prewitt** 垂直边缘模板



**sobel** 垂直边缘模板



垂直方向的边缘被增强